Abstract

BLU-ICE is a graphical Interface to the Distributed Control System for crystallographic data collection at synchrotron light sources. Designed for highly heterogeneous networked computing environments, it uses an efficient message passing protocol for communication between underlying, disparate hardware control systems and multiple, fully synchronized instances of the graphical user interface. A server process at each beam line centralizes control over all beam line hardware and provides a central point through which all messages between user interfaces and hardware control systems pass. The server also embeds a scripting engine capable of handling complex operations spanning multiple control systems with a powerful event driven language. The highly graphical BLU-ICE user interface simplifies sophisticated experimental protocols and complex motor relationships, hides the complexities of the control and computer infrastructure, and allows the user to concentrate on the problem at hand – the collection of diffraction data or the maintenance of a beam line. The design philosophy of BLU-ICE and the Distributed Control System exemplifies the Collaboratory paradigm, featuring cross-platform portability, remote access capability, and collaboration enabling features.

Distributed Control System Overview

The DCS Protocol

All components in the Distributed Control System speak the DCS protocol. The messages are sent as simple text messages via TCP/IP sockets.

BLU-ICE

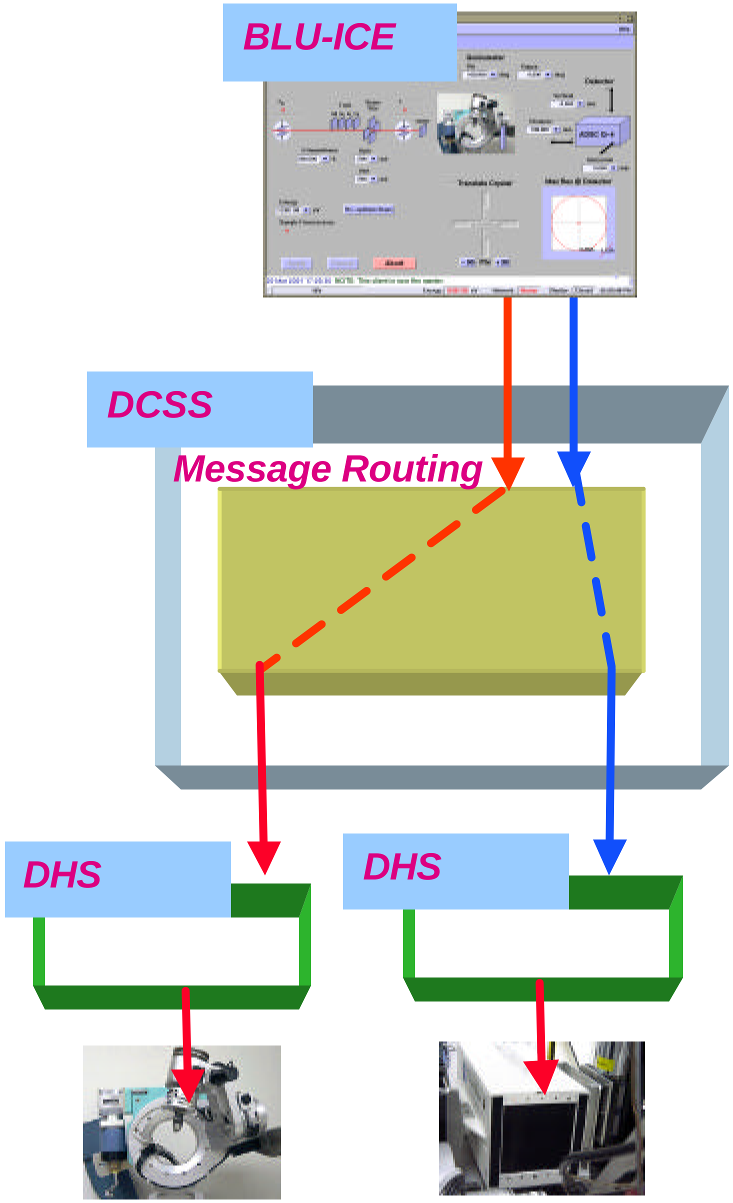
The BLU-ICE program is a client of the Distributed Control System. It provides the user with a powerful graphical interface to the system.

DCSS

The Distributed Control System Server (DCSS) is a key component of the system. It functions primarily as a message router, enabling communication between multiple GUI clients and the various hardware components.

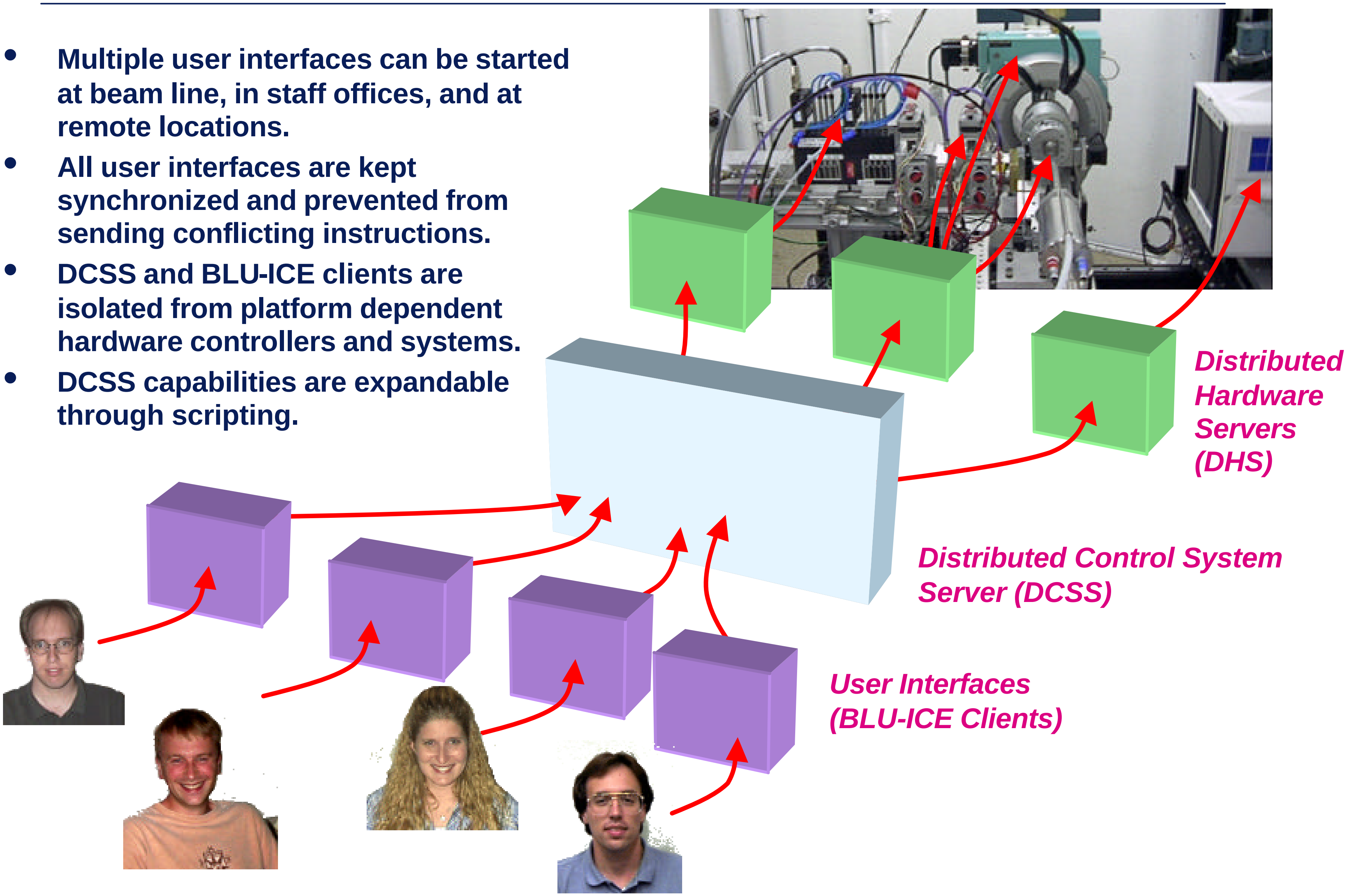
DHS

The Distributed Hardware Servers (DHS) act as translators, converting the DCS message protocol to the language of a third party hardware controller.



Benefits of the Distributed Control System

- Multiple user interfaces can be started at beam line, in staff offices, and at remote locations.
- All user interfaces are kept synchronized and prevented from sending conflicting instructions.
- DCSS and BLU-ICE clients are isolated from platform dependent hardware controllers and systems.
- DCSS capabilities are expandable through scripting.



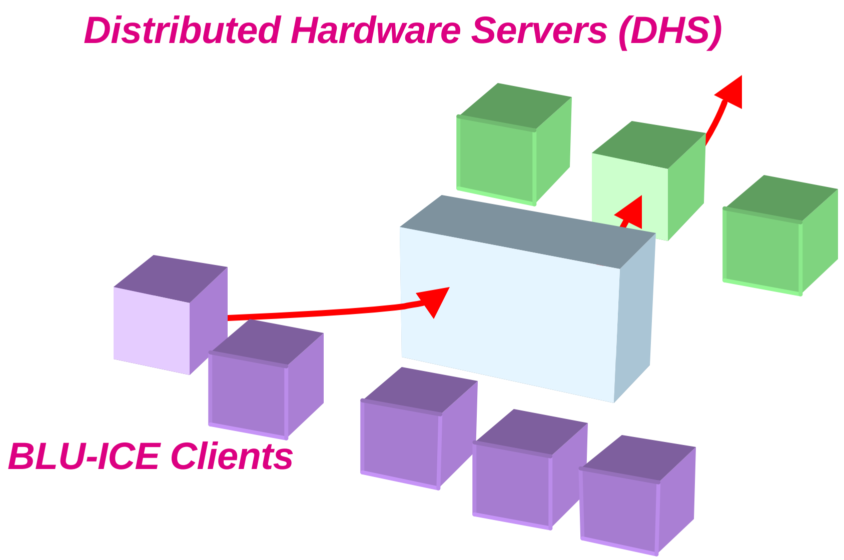
Message Passing Architecture: Moving a motor

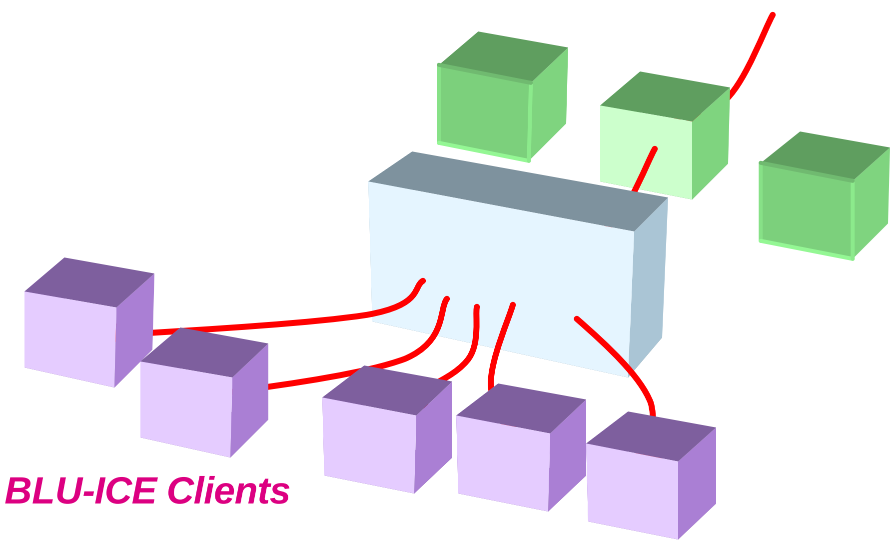
Initiating a move command

When a BLU-ICE user initiates a control command, the message must first pass through DCSS before being forwarded to the appropriate hardware server.

DHS starts the motion

DHS receives the control message and issues the command to the controller using the protocol it understands.





Monitoring hardware motion

Each Hardware Server is responsible for updating DCSS with the current position of moving motors.

Synchronizing BLU-ICE clients

DCSS will broadcast new positions for motors and changes in hardware status to all BLU-ICE clients.

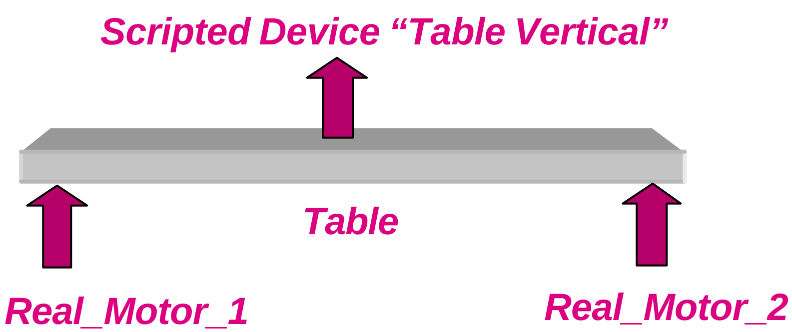
DCSS and the Embedded Scripting Engine

Scripting Engine

Embedded within DCSS is a TCL language interpreter. Also known as the Scripting Engine, this interpreter allows complex relationships for motors to be written, and complex operations to be performed. TCL's powerful event handler allows scripts to wait for certain events to occur, such as motor motion to complete, without locking the system. There are two types of scripts that can be written: "scripted devices" and "scripted operations".

Scripted Devices

A common scripted device is a motor that depends upon several "child" motors in order to move. An example would be a "table_vertical" motor, which may depend upon two independent motors, one at each end of a table. The relationship between the two motors is described by a script processed by the scripting engine.



Parent and Child Motors

In the above example, "Table Vertical" would be a parent of two child motors: "Real_Motor_1" and "Real_Motor_2". Moving "Table Vertical" would involve moving its two child motors.

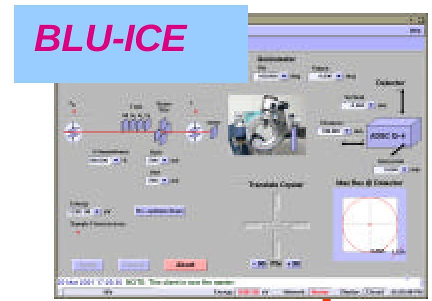
Scripted Operations

A scripted operation has access to all motors, scripted devices, and other scripted operations. An example scripted operation could be an "optimize beam" operation which would move motors to search for the location with the greatest beam intensity.

Scripting Engine Architecture: Path of a Control Message

Scripting Engine Connects to the DCSS Message Router

When DCSS is started, the Scripting Engine connects to the Message Router twice. With one connection it claims that it is a GUI client. With the other connection it claims it is a hardware server.

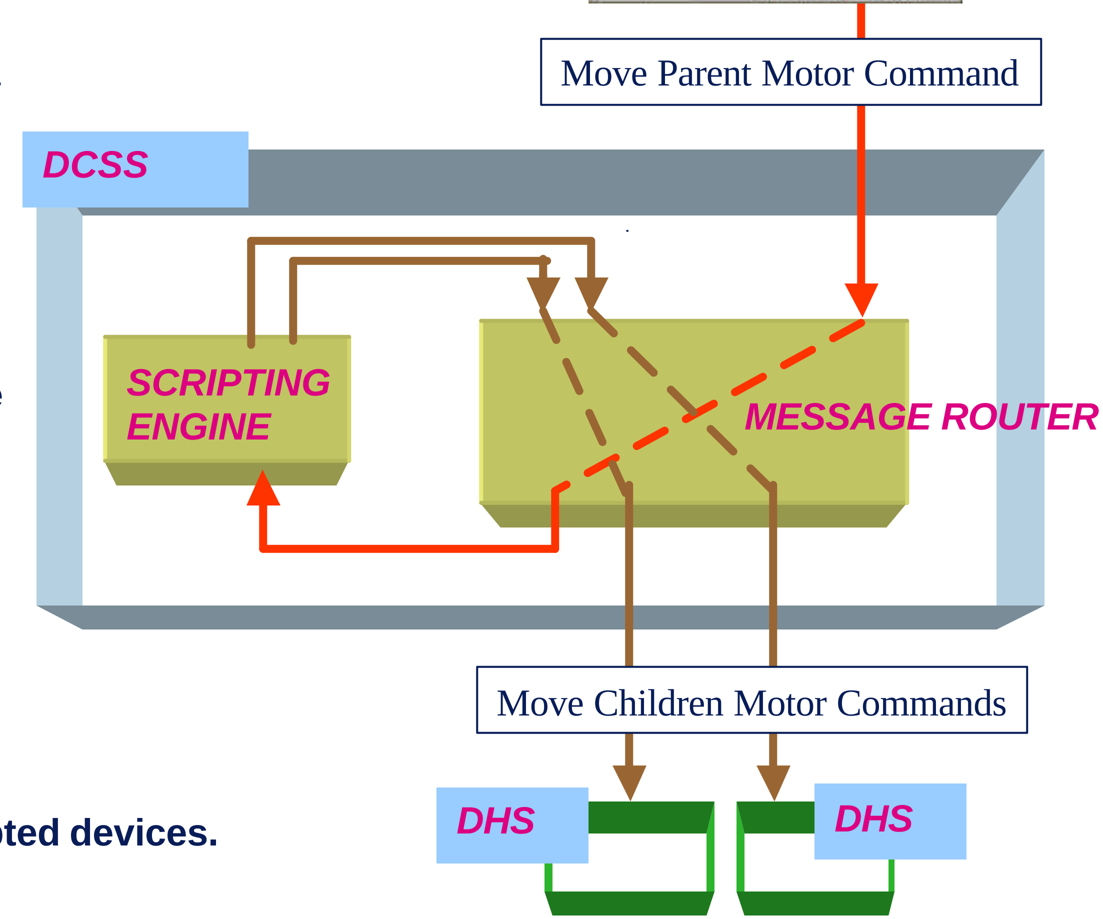


Message Routing

A request from BLU-ICE to move a scripted device is forwarded to the Scripting Engine by the Message Router.

The Scripting Engine processes the script associated with the device and issues additional move commands to the Message Router as the script dictates.

The additional move commands are forwarded to the appropriate external Hardware Servers.



Hierarchical Architecture

Children motors can also be scripted devices.

Scripted Devices: Path of a Response Message

SSRL

All BLU-ICE clients see child motors updated continuously.

The DCSS Message Router protocol automatically sends every motor position update from a hardware server to all GUI clients.

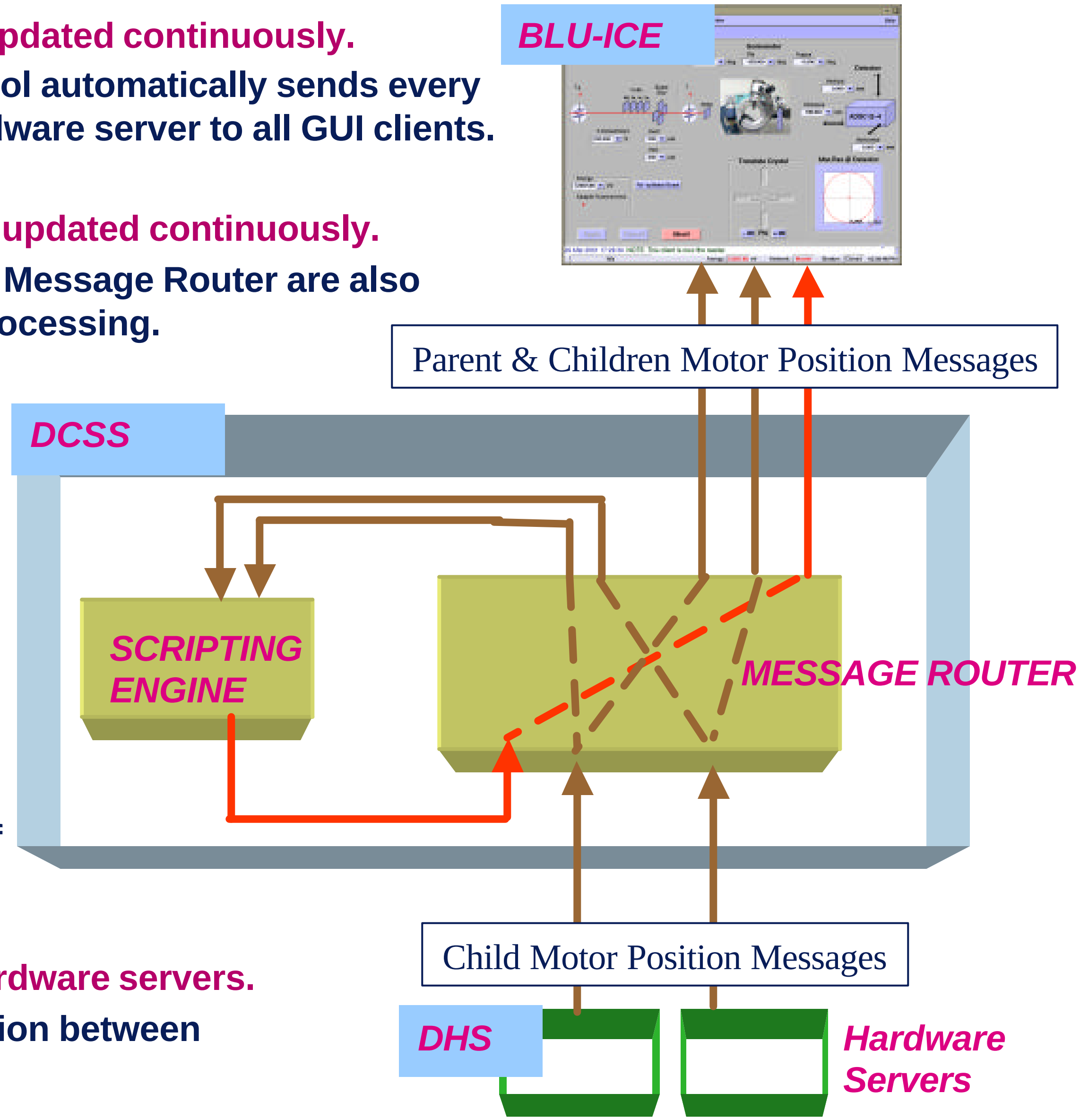
All BLU-ICE clients see parent motors updated continuously.

All motor updates received by the Message Router are also sent to the Scripting Engine for processing.

Within the Scripting Engine, the updated motor positions trigger scripts which recalculate the new position of the parent motor.

The Scripting Engine updates the Message Router with the new position of the Parent motor.

The Message Router forwards the new parent motor position to all of the BLU-ICE clients.



Scripted devices can span multiple hardware servers.

This architecture allows coordination between multiple control systems.

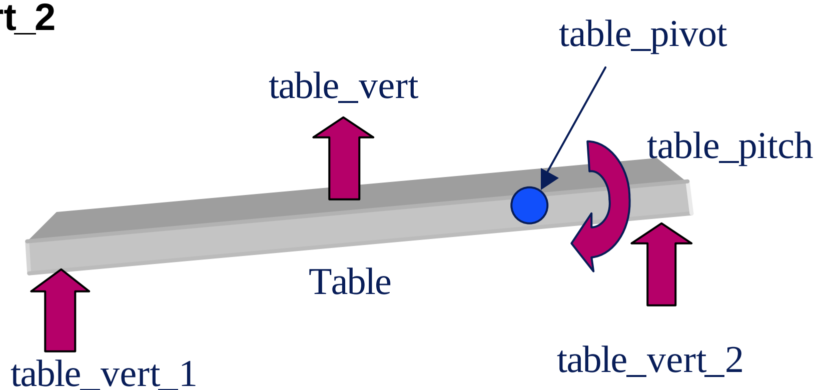
Example Scripted Device: TABLE_VERT

```
proc table_vert_move { new_table_vert } {

   # import device variables
   variable table_pitch

   # move the two motors
   move  table_vert_1  to  [ table_vert_1_calculate $new_table_vert $table_pitch ]
   move  table_vert_2  to  [ table_vert_2_calculate $new_table_vert $table_pitch ]

   # wait for the moves to complete
   wait_for_devices  table_vert_1  table_vert_2
}
```




```
proc table_vert_1_calculate { tv tp } {

   # import device variables
   variable table_pivot
   variable table_v1_z
   variable table_pivot_z

   # calculate position of table_vert_1 given vertical height and pitch
   return [ expr  $tv + ( $table_v1_z - $table_pivot_z ) * tan ( [ rad $tp ] )  ]
}
```

BLU-ICE: The Graphical User Interface

Portability and Scalability

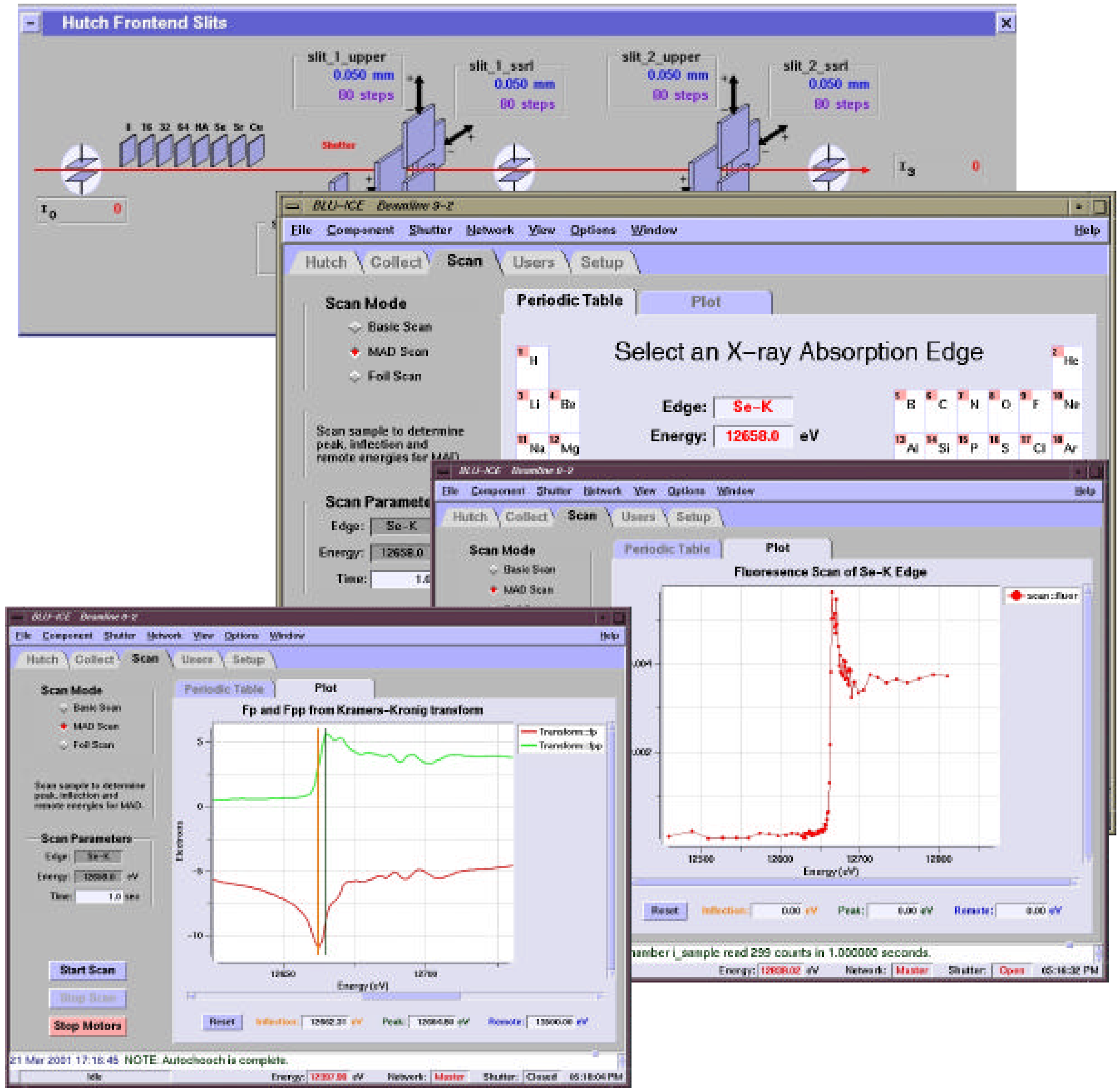
The GUI is written in TCL, the same language as the Scripting Engine, but with the addition of the TK graphical extensions. The TCL/TK language is highly portable.

Sophisticated Control

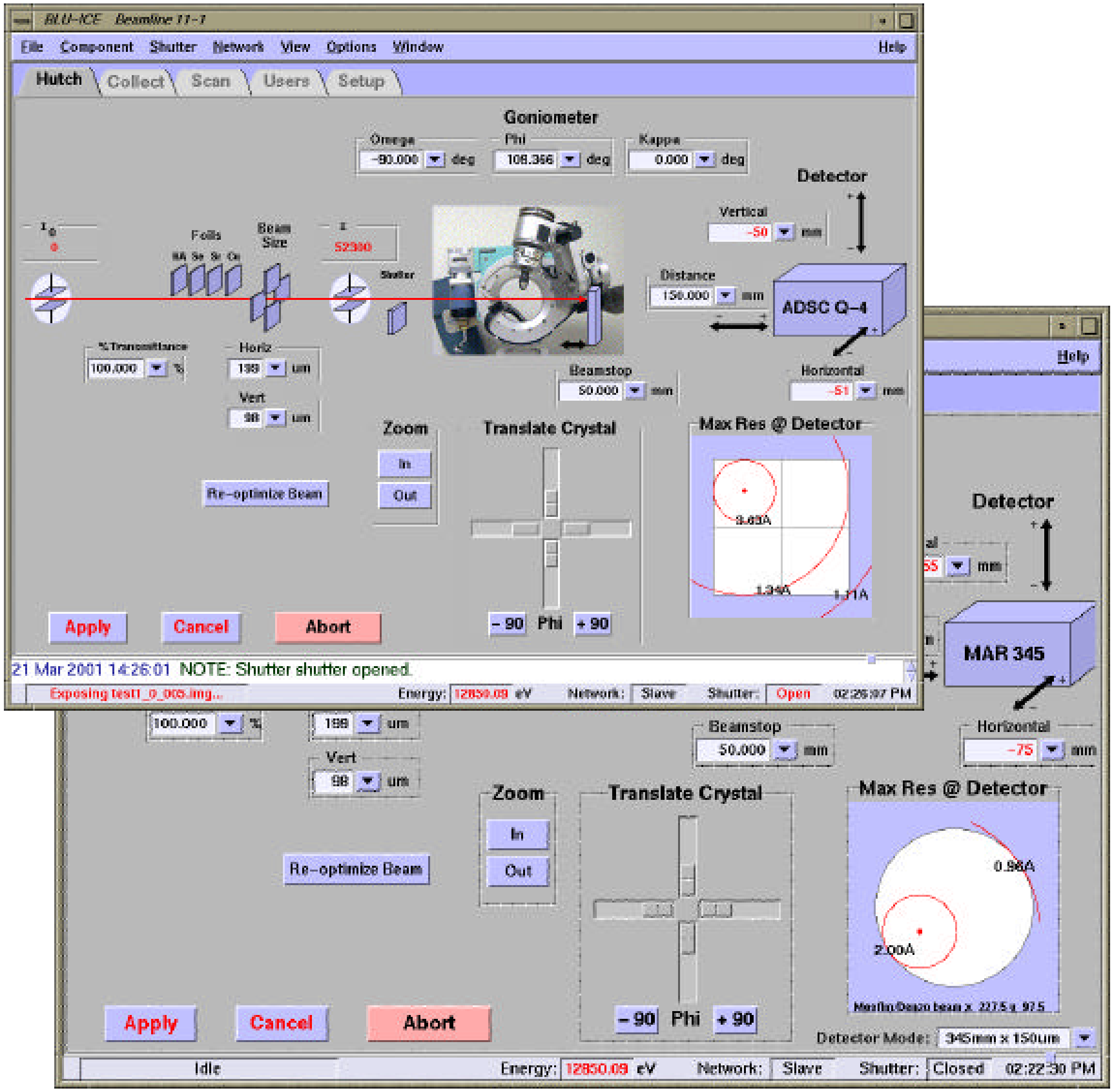
Highly graphical images simplify complex hardware and parent/child motor relationships.

Sophisticated Tools

BLU-ICE assists with common experimental procedures through interactive diagrams and graphs familiar to all scientists.



Features of the BLU-ICE Experimental Setup Window



Condensed Beam Line Status

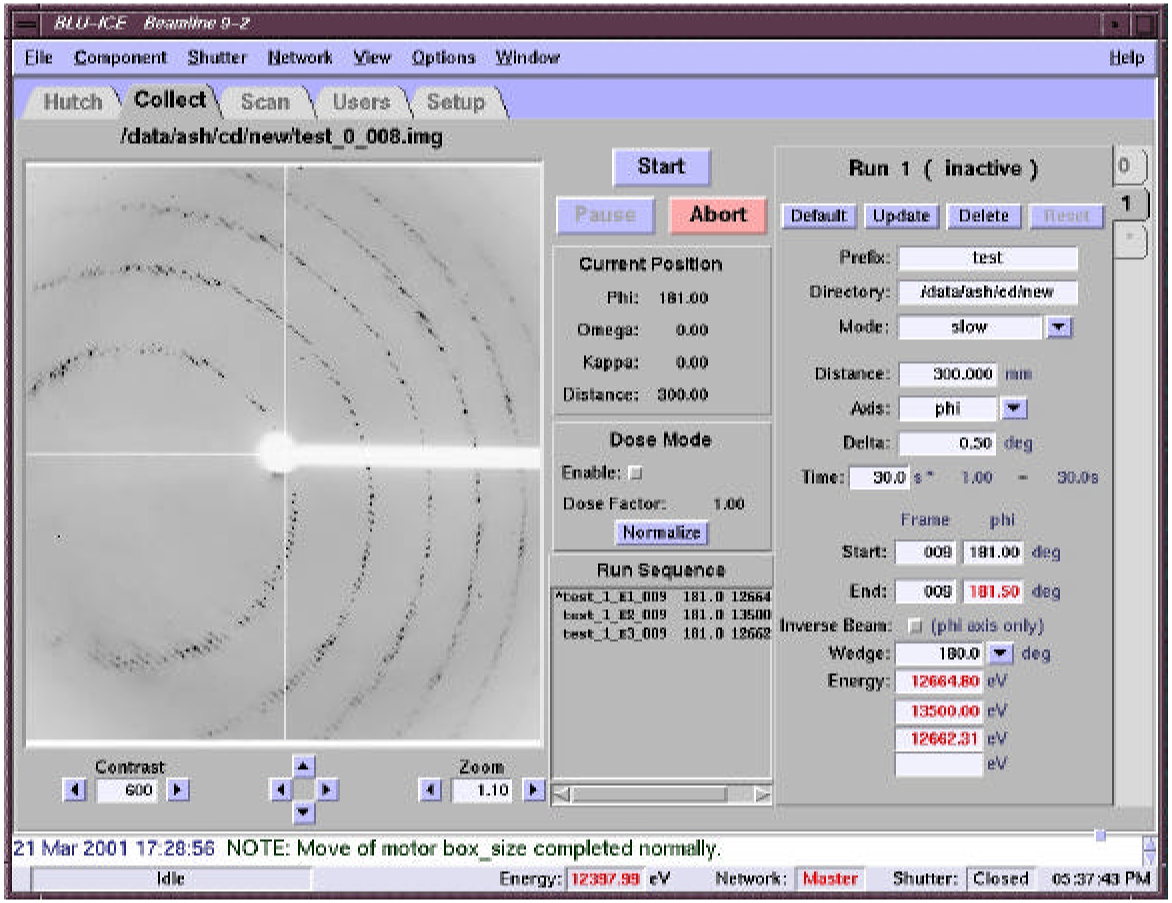
- Access to all motors pertinent to the experiment.
- Motor drop down menus allow selection of new position from allowed range of motion.
- Animated shutter and foil states.
- Different GUI variations for MAR345 and Quantum 4 CCD detectors.

Tools for Setting Up Experiment

- Optimize beam button.
- Point-and-click resolution calculator.
- Crystal centering controls.

Data Collection Control with BLU-ICE

SSRL



Generic Detector Interface

- Both Mar 345 and Quantum 4 CCD use the same data collection interface. Only the selection of the detector mode varies.

Intuitive MAD Data Collection

- Multiple runs can be defined and started together
- Supports multiple energies, inverse beam, and wedges.

Run Sequence Preview

- Lists details of data frames to be collected for each run.
- Changes as data collection parameters are edited.

Pause / Restart Capability

- Data collection may be paused at any time.
- Next frame to be collected can be selected in sequence window.

Exposure Control

- Highly intuitive dose mode displaying current exposure time.
- Multiple pass exposures

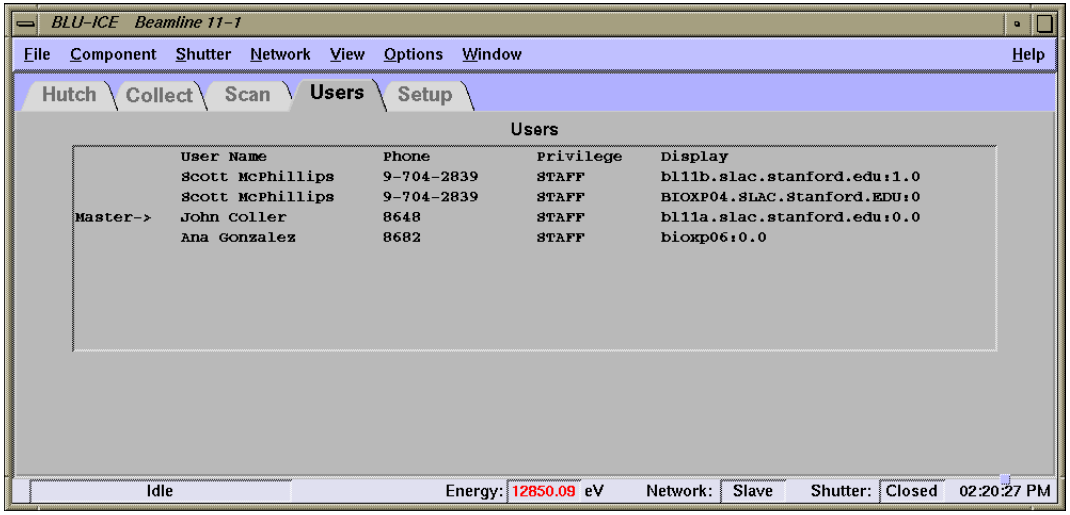
BLU-ICE Collaboration Features

Synchronized BLU-ICE Clients

All information regarding motor positions, shutter states, run definitions, and data collection status remains up-to-date on all BLU-ICE clients.

BLU-ICE Not Critical

A BLU-ICE client may be closed and reopened without affecting ongoing data collection.



Security

Only users enabled by staff may connect to the DCSS beam line server.

Visible Collaborators

BLU-ICE displays a list of other BLU-ICE clients, who started the client, and from what computer and display it was started on.

One BLU-ICE in Control

Each BLU-ICE client has the ability to request control of the beam line. However, DCSS limits the number of controlling clients to one. This prevents conflicting instructions from being issued by two different users.

BLU-ICE Beam Line Configuration Window

Graphical Depiction of Beam Line Hardware

- Diagrams of optical elements such as mirrors, monochromators, and slits
- Pictures of hutch equipment

Configuration of Devices

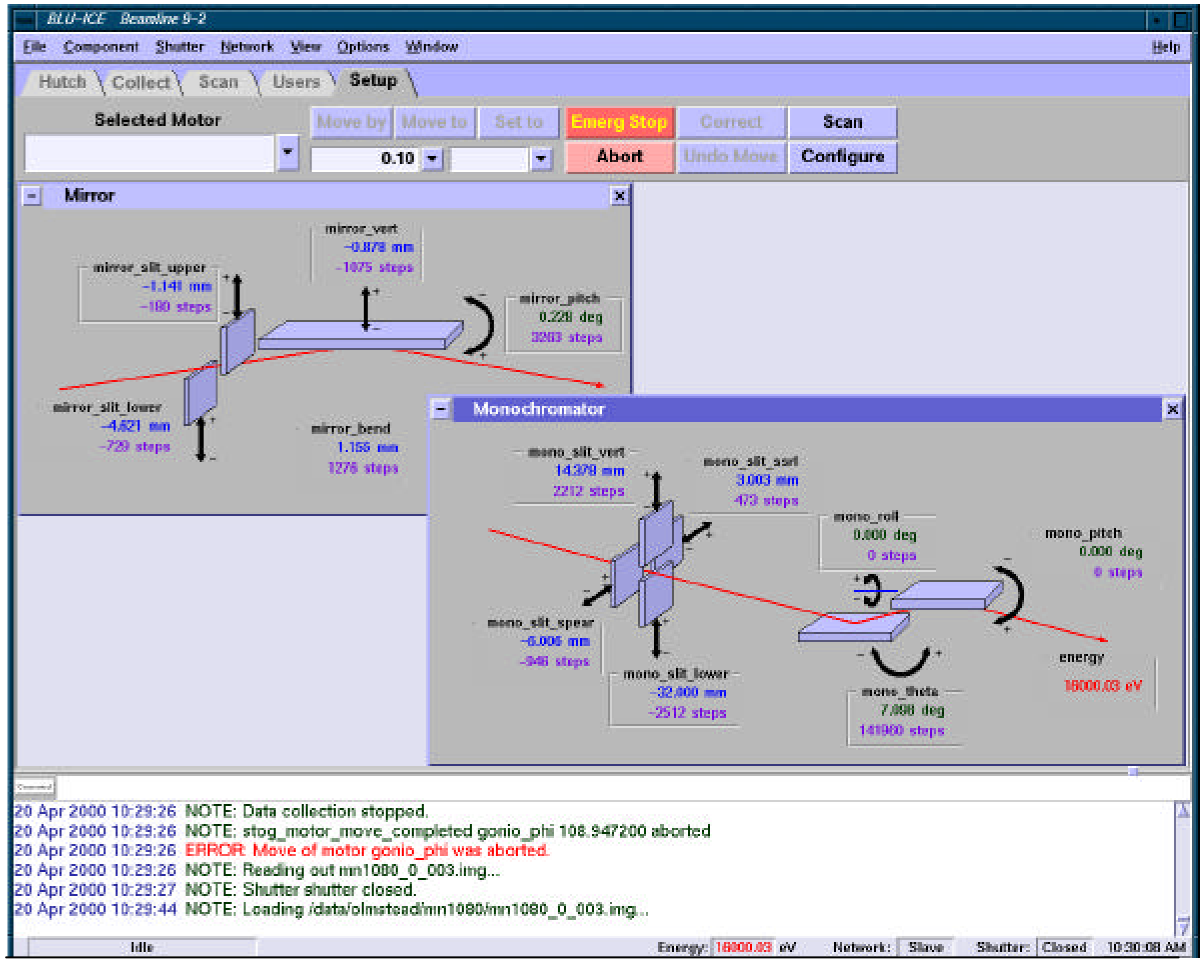
Stepper and DC servo motors' speed, acceleration, backlash, scale factors, etc, may be modified on the fly using the motor configuration windows.

Configuration of Beam Line

Complete beam line configuration can be saved and restored at will.

Limited Access for Users

The configuration features are only accessible by staff.



Distributed Control System Status & Summary

BLU-ICE in Production Phase

- Currently runs on three beam lines at SSRL: BL9-2, BL9-1, and BL11-1.
- All beam lines have similar interface, reducing time to train users.

Supported Platforms

- BLU-ICE can be run on any platform with TCL/TK (UNIX and Windows).
- DCSS and the DHS are compiled in C/C++ using a cross platform library. The cross platform library wraps system calls with a simple API and compiles on Digital Unix, IRIX, OPENVMS, Windows NT, and Linux.

Subset of Currently Supported Hardware

Controller	Function	Comments
DMC2180	Motion Controller	Ethernet Based Controller from Galil Motion Control
SSRL Control System (ICS)	Control System	DHS wraps the complete system
MAR 345	Detector	DCS protocol includes a generic detector interface
Quantum 4 CCD	Detector	

Extendibility (Scripting Language Support)

Extending the capabilities of DCSS is done by writing scripts in TCL (Tool Command Language). TCL provides control structures, string manipulation, file I/O, socket I/O, lists, arrays, and events.

BLU-ICE: The Future

Enhanced Security for Remote Access

Each scripted device, motor, and scripted operation will have a configurable parameter which will enable or restrict remote access.

Automatic Crystal Centering

Image analysis and automatic loop centering is currently under development at SSRL. This will be available to all users through BLU-ICE.

Video Feeds

BLU-ICE will have embedded images displaying live video from multiple cameras installed at the beam lines.

Automatic Indexing and Data Collection Strategy

Additional tools will be developed and integrated into the BLU-ICE package for rapid setup of diffraction experiments and determination of cell parameters.

Quantum 315 CCD

The Quantum 315 CCD will be supported in the near future.